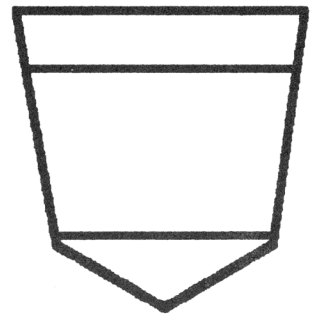


POCKET COMPUTER NEWSLETTER



© Copyright 1985 POCKET COMPUTER NEWSLETTER

Issue 37 - January

REVIEW OF THE SHARP PC-1350

The new Sharp PC-1350 is becoming available in the United States now. It is somewhat trimmer in size than the PC-1500 (about 3/8"ths of an inch less in length and width, and 1/4 inch less in thickness). It is equipped with a protective "half-shell" (in the same fashion as the PC-1261). When the unit is not in use, the shell protects the keyboard/display surface. When utilizing the PC, the shell is reversed and mounted on the bottom of the unit for safekeeping. Shell and all, the new PC measures approximately 7-1/4 inches in length, by 3 inches in width, and a bit of 3/4 of an inch in thickness. It is slim enough to fit in a typical shirt pocket, but its length would cause it to stick out somewhat if carried there.

The PC-1350 weighs in at about 9 ounces, quite a bit lighter than the 1500. Much of this decrease in weight is due to its lack of the 4 type AA batteries used in the PC-1500. These have been replaced by a pair of lithium cells (type CR-2032) that are said to be capable of powering the PC for approximately 250 hours of "normal" use.

The most striking feature of the new PC is its large, 4-line liquid-crystal (LCD) display. The display window is one by four inches in size, and occupies the upper-left portion of the keyboard surface when viewed from the normal operating position. The active pixel area within the display window measures 3/4 by 3-1/2 inches. It contains 150 columns and 32 rows of display elements. Individual dots (pixels) may be controlled to provide graphics. In the text mode, the pixels are used to form characters in a 5 by 7 matrix. Using such a matrix permits four lines of 24 characters-per-line to be displayed at one time. A contrast control on the right side of the PC gives a user the ability to adjust the display for comfortable viewing under various lighting conditions. A number of annunciators appear along the left border of the pixel matrix. These annunciators serve as mode and status indicators.

There are three rows of keys directly below the display. Two of these contain 11 keys, the third contains 10 with the ENTER key occupying the equivalent of two key positions. These keys represent the 26 letters of the alphabet, plus the SHIFT, DEF, SML, SPC, = and ENTER operations. The SML key permits selecting upper and lower case letters. The SHIFT key is used to provide alternate characters for some of the keys. These alternate characters consist primarily of punctuation marks (such as: ! " # \$ % & ?) and special mathematical operators (including < and >).

The right side of the keyboard holds a 4 by 5 matrix of keys (larger than the alphabetical keys) containing the numbers 0-9, the four primary mathematical operators (+, -, *, and /) and commonly used punctuation (period, comma, colon, semicolon and parenthesis). Above the 4 by 5 matrix are ten more keys used for controlling the display, selecting modes, etc., and a sliding on/off switch.

One marked improvement (over the PC-1500) in regards to the keyboard is that commonly used editing operations no longer require the use of the SHIFT key. For instance, you can insert and delete characters by simply pressing the INS and DEL editing keys. Furthermore, the punctuation marks that are frequently used in programming (colon, semicolon, comma and parenthesis) have their own keys and do not require the use of the SHIFT key. This considerably eases the task of keying in BASIC programs.

The PC-1350 contains about 3K of user memory (RAM). This does not seem like much for a "third-generation" PC. But, you can expand memory by installing a thin RAM-card in the back of the unit. A CE-201M card provides 8K of additional RAM. A model CE-202M card gives an additional 16K. However, these battery-backed expansion memories are not cheap. Buying a 16K card will almost double the basic price of a PC-1350 at the present time. There are, however, some nice features about these memory cards. First, they can maintain programs and/or data when removed from the unit. (Unlike the modules for the PC-1500 which were designed strictly for the storage of programs when removed from the PC.) The structure of these modules is such that you can switch programs and data bases by swapping in a new card (and without performing a lot of fancy procedures). You can elect to have the 3K of user memory within the PC-1350 tacked onto the end of the memory in the RAM card (and thus use it for variables storage) or to be ignored. The former option means you can use the memory card primarily for program storage (while using the 1350's RAM for data). The latter means you can install a complete self-contained software package (program and data) within a card and not be concerned about the boundary between the card's memory and that of the 1350's. (Software developers take note! This means you can develop complete packages without worrying about the types of complications that could appear when using expansion modules in the PC-1500!)

The PC-1350 contains 40K of ROM containing essentially the same type of BASIC interpreter as in the earlier Sharp 1250/1260 series. However, instead of features such as the "help" directory and "automatic calculations" found in the PC-1260/1261, the 1350's extra ROM is used to control the large display and provide capabilities for its extra input/output port.

What extra input/output port? Well, in addition to the port that is used to communicate with a CE-126P printer and cassette interface, the PC-1350 contains a second "serial communications" port! Yes, on the right hand end of the PC-1350 is a 15-pin connector that enables the 1350 to communicate with external devices. This port is designed to operate at logic levels of 0 and +5 volts. The port is programmable in BASIC and such parameters as: the baud rate (300 to 1200), number of bits-per-word, parity, number of stop bits, etc., can be set under program control. Stated another way, the PC-1350 has a great deal of the capability

of the earlier CE-158 communications interface built right into it! (This built-in communications capability is one reason the PC-1350's price is not quite as low as some people might have expected a third-generation PC to be.)

At the operational level, the PC-1350 is a lot more "comfortable" PC to use than previous Sharp versions. It is apparent that Sharp has been doing its homework and listening to users to improve its products. Many of the nice touches are subtle but important. Just to give an idea of the kind of user-friendly type of operation implemented in this machine, consider how the PC-1350 handles a simple PRINT statement that contains a multi-line string of text. It utilizes automatic "word-wrap" so that when the PRINT statement is executed, the text is easy to read! Note that this is the BASIC interpreter itself that is doing the word-wrapping on the fly. The programmer does not have to be concerned about how a sentence will appear on the 4-line LCD. (Unless the programmer wants to over-ride word wrap. This is easy to do. Just use the CURSOR statement.)

Of course, having the big 4-line display on the PC adds to its general ease of operation. You can usually view most of the information on a program line (as the four lines show a total of 96 characters at a time). This, coupled with the single-function editing keys (character insert and delete), and the fact that punctuation commonly used in programming can be invoked without using a SHIFT key, makes the inputting of programs through the keyboard somewhat easier than on previous Sharp pocket computers.

As for speed of operation, the PC-1350 benchmarks at approximately the same rate as the PC-1500. Preliminary research indicates, however, that the PC-1350 has a different CPU than that used in the PC-1500. In fact, it appears to utilize the same type of machine codes as that utilized in the Sharp PC-1250 & PC-1260 series. If this

proves out to be true, it could be interesting for people who like to program at the machine language level.

All-in-all, the new Sharp PC-1350 seems to offer significant potential. The display is large enough to be of practical value, say, in a light word processing (memo-taking) environment. It is possible to envision, for example, this little PC being connected (via the serial port) to a full-sized keyboard or a combination keyboard and printer. That same serial port could also be used for uploading and downloading files to some sort of mass storage device or a desktop computer. Placing a 16K RAM card in the unit gives close to 20K of user memory. That is enough to handle quite a few types of practical applications. While the RAM cards are currently rather expensive, the price may drop as volume picks up. Even at their current prices, many professionals may find it beneficial to keep their most frequently used application programs on the cards, ready for virtually instantaneous use.

If Sharp is planning a fourth-generation PC, they might want to consider providing an operating system that would permit manipulating multiple files in memory at one time. The capabilities of the Hewlett-Packard handhelds, such as the HP-71B, in this regards, have much to recommend them. Pocket computers are especially valuable in environments that call for many types of relatively small tasks to be performed. Switching between those tasks is currently the biggest chore. The Sharp method of merging files (so that multiple programs can be executed but not edited) and its limited ability to handle files of data puts it at a distinct disadvantage in the hotly competitive PC market. (The HP-71B clearly has then beat in this regards.) However, at less than half the price of a HP-71B, the new Sharp PC-1350 can certainly be considered a bargain.

— FOR PC-1250/51/60/61 USERS —

PAYROLL PROGRAM FOR 1250/51 AND 1260/61 POCKET COMPUTERS

Here we are at the start of a new (alas, taxable) year. Seems like a good time to update our old Payroll program to take account of the new (higher) FICA tax rate and adjusted withholding rates. The original version of our Payroll program appeared way back in Issue 03 of *PCW* and a modified version, reorganized thanks to the efforts of *David Notto*, was published in Issue 05.

This updated version has been slightly customized for the PC-1250/51 (as well as PC-3) and the PC-1260/61 models. A variation for lines 40 - 50 is provided as an alternate means of programming. (Why not try both methods? Keep the one you like best!)

If you run a small business, the program can ease the weekly chore of calculating payroll deductions. Alternately, if you are employed by someone else, you can get a close approximation of what your employer should be withholding. This program is based on widely published tax-withholding tables. However, there can be variations of a percent or two between various withholding methods. (Consult your personal tax-advisor for tax advice. *PCW* provides this program without warranty of any kind and is not engaged in the practice of issuing tax advice.)

Program Payroll Deductions.

```
10:"S" USING "#####.##"
   :H=100
20:G=0: INPUT "ENTER GR
   OSS PAY? ";G
30:E=0: INPUT "NR. OF E
   XEMPTIONS? ";E
40:WAIT 0: PRINT "SINGL
   E(S)/MARRIED(M)? ";M
   $= INKEY$ : IF M$=""
   THEN 40
45:IF M$="S" LET M=0:
   GOTO 60
50:IF M$="M" LET M=1:
   GOTO 60
55:BEEP 1: GOTO 40
60:WAIT :X=0:T=G-20*E:
   GOSUB 600+200*M
70:F=.0705*G
80:X= INT (H*X+.5)/H:
   PRINT "FWT= ";X
90:F= INT (H*F+.5)/H:
   PRINT "FICA= ";F
```

```

100:Z=0: INPUT "OTHER WITHOLDINGS? ";Z
110:Z= INT (H*Z+.5)/H
120:G=G-X-F-Z: PRINT "NET PAY= ";G
130:GOTO 20
600:IF T>=663 LET X=150.50+.37*(T-663): RETURN
610:IF T>=556 LET X=114.12+.34*(T-556): RETURN
620:IF T>=440 LET X=79.32+.30*(T-440): RETURN
630:IF T>=292 LET X=42.32+.25*(T-292): RETURN
640:IF T>=185 LET X=21.99+.19*(T-185): RETURN
650:IF T>=84 LET X=6.84+.15*(T-84): RETURN
660:IF T>=27 LET X=.12*(T-27)
670:RETURN
800:IF T>=897 LET X=195.75+.37*(T-897): RETURN
810:IF T>=684 LET X=125.46+.33*(T-684): RETURN
820:IF T>=578 LET X=95.78+.28*(T-578): RETURN
830:IF T>=472 LET X=69.28+.25*(T-472): RETURN
840:IF T>=384 LET X=49.92+.22*(T-384): RETURN
850:IF T>=192 LET X=17.28+.17*(T-192): RETURN
860:IF T>=48 LET X=.12*(T-48)
870:RETURN

```

Alternate Method for Lines 40 - 50

```

40:WAIT 0:M=0: INPUT "SINGLE(S)/MARRIED(M)? ";A$
45:IF A$="S" LET M=0: GOTO 60
50:IF A$="M" LET M=1: GOTO 60

```

Program Memory Dump.

```

10:DIM A$(0)*16: INPUT "STARTING ADDRESS? ";A
15:INPUT "ENDING ADDRESS? ";G: PRINT = PRINT
20:A$(0)="0123456789ABCDEF"
25:WAIT :W= INT (A/4096)
26:X= INT ((A-(W*4096))/256)
27:Y= INT ((A-(W*4096)-(X*256))/16)
28:Z= INT ((A-(W*4096)-(X*256)-(Y*16)))
29:PRINT MID$(A$(0),W+1,1); MID$(A$(0),X+1,1); MID$(A$(0),Y+1,1); MID$(A$(0),Z+1,1)
30:WAIT 0: FOR B=0 TO 7 :C=( PEEK (A+B)) AND (&F0):D=1+ INT (C/16):E=( PEEK (A+B)) AND (&0F)
35:IF B=7 WAIT
40:PRINT MID$(A$(0),D,1); MID$(A$(0),E+1,1);" ";
45:NEXT B
48:F= INT (A/64):A=A+8: IF A>G THEN STOP
49:IF INT (A/64)<=F THEN 30
50:GOTO 25

```

MEMORY DUMP PROGRAM

Wow! User interest and excitement in the Sharp PC-1260 & PC-1261 models is really heating up. The discovery that the models contain the undocumented commands PEEK and POKE has lead to a new round of detective work on the CPU instruction set. Wouldn't you know it, urged on by such pocket computing pioneers as Norlin Rober (of PC-1500 fame), others have been rapidly cracking the machine code used in the newer PCs. (It appears that the PC-1260 and PC-1261, as well as possibly the new PC-1350, use the same CPU as the 1250/51.) We will have detailed reports of this ongoing work in future issues of PCN.

Meanwhile, if you want to do some exploring on your own, we have adapted a small memory dump program (from Issue 26 of *PCW*) for use on 1260/1261 units. The adaptation is from a routine originally crafted by *Marlin Rober* for use on the PC-1500. A few "bells and whistles" have been added to make it a little easier to use. Note that line 15 contains a "PRINT=" statement. Set this statement to read PRINT=PRINT if you want the dump to the liquid-crystal display. Set it to read PRINT=LPRINT if you want the dump listed on a CE-125 or

CE-126P printer.

When you start this memory dump program, it will ask for starting and ending addresses. You can enter these in decimal format. Alternately, you can work in hexadecimal by preceding the values with the ampersand (&) sign.

If you are using the LCD for a dump, you need to press the ENTER key after each line of information has been displayed. Note that the memory is dumped in blocks of 64 hexadecimally-formatted bytes. Each group is preceded by the starting address (using hexadecimal notation) of that group.

When using the printer, the dump proceeds automatically

until the block requested has been outputted. Groups of 64 bytes are separated by a line containing an address value.

You can use this memory dump program to start exploring your PC. On the PC-1261 it appears that some RAM is located starting at decimal address 16384 (68000 hexadecimal). Formulas assigned by a user for use with the "automatic calculation mode" appear to be stored here.

This memory dump program should be readily adaptable to the PC-1250/51 models. Remove the PRINT= statement in line 15. Change the PRINT statements to LPRINT if you want to use a printer. (These alterations will be required on a 1250/51 since it does not support the PRINT= command.)

FOR HEWLETT-PACKARD HP-71B USERS

Program Payroll Deductions.

```
10 DELAY 8, .5 @ WIDTH 96 @ DESTROY ALL @ FIX 2
20 A$=KEY$ @ INPUT "ENTER GROSS PAY: ", "0";G
30 INPUT "NR. OF EXEMPTIONS? ", "0";E
40 INPUT "(S)INGLE/(M)ARRIED? ", "S";M$
50 IF M$="S" OR M$="M" THEN 60 ELSE BEEP @ GOTO 40
60 X=0 @ T=G-20*E @ GOSUB M$
70 F=.0705*G
80 PRINT "FMT= ";X
90 PRINT "FICA= ";F
100 A$=KEY$ @ INPUT "OTHER WITHHOLDING? ", "0";Z
110 G=G-X-F-Z
120 PRINT "NET PAY= ";G
130 GOTO 20
600 'S': IF T>=663 THEN X=150.50+.37*(T-663) @ RETURN
610 IF T>=556 THEN X=114.12+.34*(T-556) @ RETURN
620 IF T>=440 THEN X=79.32+.30*(T-440) @ RETURN
630 IF T>=292 THEN X=42.32+.25*(T-292) @ RETURN
640 IF T>=185 THEN X=21.99+.19*(T-185) @ RETURN
650 IF T>=84 THEN X=6.84+.15*(T-84) @ RETURN
660 IF T>=27 THEN X=.12*(T-27)
670 RETURN
800 'M': IF T>=897 THEN X=195.75+.37*(T-897) @ RETURN
810 IF T>=684 THEN X=125.46+.33*(T-684) @ RETURN
820 IF T>=578 THEN X=95.78+.28*(T-578) @ RETURN
830 IF T>=472 THEN X=69.28+.25*(T-472) @ RETURN
840 IF T>=384 THEN X=49.92+.22*(T-384) @ RETURN
850 IF T>=192 THEN X=17.28+.17*(T-192) @ RETURN
860 IF T>=48 THEN X=.12*(T-48)
870 RETURN
```

PAYROLL DEDUCTIONS PROGRAM FOR HP-71B

This program is a customization of the Payroll program (described for other models of PCs in this issue) specifically for the HP-71B. [Refer to other articles in this issue for additional background information concerning this program.]

The program takes advantage of the 71B's advanced capability in handling inputs. For instance, the ability to "cue" a default input value. Using this capability means that default values do not have to be set up by using a separate "let" statement. Thus, for example, in line 20 of the program, instead of preceding the input statement with a statement setting C=0 (as a default value), that default value is made a part of the INPUT statement. You might want to customize this default value to your own application. For example, if you have a number of employees earning \$350.00 gross per week, you could set the default value to that figure (instead of the 0 used in the program listing).

This HP version also utilizes labels in the tax lookup table routines (beginning at lines 600 and 800). Observe how line 60 obtains the label of the appropriate lookup table from the string M\$ (which is inputted by the user in response to line 40).

If you are a newcomer to the use of the HP-71B, then you may want an explanation about the use of the statements A\$=KEY\$ in lines 20 and 100. As you may be aware, the -71B has a lot of features, including a "type-ahead" input buffer. This capability allows a user to start entering data before a prompt even appears on the screen. However, sometimes this advanced capability can trip a new programmer up! For instance, when using a delay of infinity (which is what you have when the line-scrolling parameter in a DELAY statement is 8 or more, as in this program), the use of the END LINE key causes that character to be stored in the type-ahead buffer.

Whatever is in the type-ahead buffer is then used as the data for the next INPUT statement. That means, the next INPUT statement picks up an END LINE character as its initial character, thereby terminating the input operation and effectively causing the input to be a "null" character. That is usually a rather undesirable manner of operation. One way to get around this situation is to effectively empty out the type-ahead buffer prior to executing an INPUT statement. The use of the KEY\$ statement does just that! Thus, the simple implied let statement used in lines 20 and 100 (A\$=KEY\$) uses a "dummy" variable (A\$) as a means of dumping out the type-ahead buffer prior to using the INPUT statements in those lines. If you find this procedure difficult to understand, just try operating the program a few times with those statements removed and watch what happens!

Be sure to make use of the line editing capability of the HP-71B when building the lookup tables that start at lines 600 and 800. Once you have entered line 600, just use the FETCH command to get it back. Then, use the cursor controls to change the line number and modify the remainder of the line appropriately. You should find this method somewhat faster than having to type in each and every line in its entirety!

If you enter the program exactly as shown in the listing, a CAT should indicate that it is taking up 789 bytes.

For some fun, test the program by entering an amount of \$5000.00 for hypothetical weekly wages, assume 4 dependents, and that you are married. You should get a FWT value of 1684.26, FICA withholdings of 352.50. With no other withholdings, the net pay would be 2965.24. If everything works out to these figures, then you can figure that at least the heart of the program is working O.K. (However, you should recheck all your table entries to make sure they are accurate before utilizing the program for serious purposes.)

FOR PC-1500 & PC-2 USERS

PC-1500 POTPOURRI

The two programs presented in the PC-1250/51/60/61 section of this issue, can, of course, be adapted to run on the PC-1500 with relative ease.

Rather than present a specific program for the PC-1500 in this issue, we are going to use this issue's column space to discuss several matters that are frequently raised by readers.

We often receive inquiries from program developers who want to safeguard their programs. "How?", they ask, "Can I lock up a program so that it cannot be listed, edited or saved on a cassette?" In other words, how can they "protect" a program so that they can sell it while preventing others from copying their work.

First of all, it is unlikely that any technique one might try to apply could be 100 percent effective. No matter what you do, the chances are that somebody else, having similar knowledge and skills, can undo your work. Thus, protection really becomes more of a game between the developer or "code maker" and the prospective "code breaker" or unauthorized duplicator. In the final analysis, it becomes a matter of wits and perseverance, knowledge and skill. For instance, virtually anyone who reads this article, is going to have knowledge about the techniques that are discussed and can use that information for or against copy protection. The information contained in *PCW* may be disseminated to perhaps five percent of all pocket computer users. Does that mean that 95% of PC users won't know about the techniques discussed here? Not necessarily. It is really impossible to determine how effective a technique might be since the

"game" is really between those that "know" and those that do not! And, there is no way of knowing who has or who does not have the requisite knowledge.

One of the first things to decide when considering copy protection is just what is it you plan on protecting. Do you want to prevent duplication of the program and unauthorized distribution? Or, do you want to protect the algorithms or program methodologies? If your primary interest is the former, then you should consider the standard legal avenues such as copyright and trademark protection along with any technical anti-duplication techniques you might apply. If you are attempting to guard algorithms or methodologies, then your tack may be quite different. Here, you may want to concentrate on making it difficult for someone else to figure out just exactly how you performed a particular operation. (Remember, while a copyright can protect a particular implementation of a program, it cannot stop someone else from using the same methodology or idea. A simple re-write of your program using different variable names, labels, text and some re-arrangement of the materials is all that is needed to legally circumvent the copyright laws.)

If you want to prevent outright duplication of your program, then you had better not plan on publishing it on magnetic tape. Anyone with a duplicating tape recorder unit can duplicate what is on a magnetic tape. If someone is serious about distributing your software (say, in a foreign country), all they need is one copy of your program recorded on tape and they are in business. Presto! They can be making copies in a matter of seconds. If someone is going to break

the copyright laws in this manner (assuming your work has been copyrighted), it is certainly not hard to do from a technical viewpoint. Remember, we are talking here about tape-to-tape copying. The program never has to be loaded into a computer! No amount of protection within the program or computer is going to be able to prevent this simple outright duplication of the information contained on the tape cassette! By placing their own labels on the copied cassettes, program pirates can easily end up selling your program (particularly in a foreign country) with little chance of being detected.

OK, you say, you are going to distribute the program on a memory module (such as the CE-160) instead of tape cassette. Now what can you do? Well, the CE-160 programming system has an option that is said to prevent the listing, editing or saving of the program. Just select that option when you place a program into the module! Does that mean your program is now safe from duplication? Hardly.

Oh sure, it means the casual user cannot simply invoke a LLIST to view the program. Nor can that user invoke a CSAVE command to make a copy on a tape recorder. However, anyone familiar with machine language techniques (such as a reader of *PCW*) would be capable of accessing the information in the module. A simple machine language memory dump could be used to obtain the contents of the module. The information obtained from such a memory dump could be loaded into another PC without the protection. Once such a copy had been made, the program could be viewed, edited and duplicated. From there, mass duplication would be no more difficult than simply making copies on a magnetic tape cassette duplicator.

So what is the poor, defenseless, pocket computer programmer to do? Well, for one, stop being paranoid about duplication per se. Believe this: if your program is so valuable that people want to duplicate it for the sake of being able to generate some "underground" sales, they are going to find a way of doing it regardless of what you do. The only salvation is that you will probably make some money too, since you will have the lead in terms of promotion, supply, supporting materials, etc. Remember, however, that the outright duplicator (copier) who distributes your copyrighted material without your permission, is breaking the law. In other words, they are viewed as crooks. Thus, they take a certain amount of risk. And, if you can prove that they are duplicating your material, you may be able to put them out of business.

Frankly, you would probably be better off to assume that there is very little, from a practical viewpoint, you can do to stop someone bent on being a crook. It is better, perhaps, to be more concerned about slowing down someone who wants to legitimately compete with you by "re-writing" your program. Remember, a good many people do not want to risk being caught and branded as outright thieves. They might, however, view "competing" with you as quite another matter. They might even define "competing" as simply doing what you have done a little differently or a little better. All they need in order to provide this legal competition is to figure out how your program operates. In other words, they need to know how you did what you did.

There are things you can do to make this difficult on their part. One thing is to put key sections of your program in machine language. Once placed in the PC as plain object code, your competition is going to have to spend a lot of time figuring out what you did. First, they have to find out where you stored the key sections in memory. Then, they will need to disassemble your code and study it in detail. This process could take months. It is subject to various kinds of interpretation and error. (You might encourage misinterpretation by deliberately inserting nonsense sections in your code.) It may involve so much work that the perpetrators may find it easier to simply design their own version of your program from scratch. Or, they may decide it is not worth competing with you!

But, will it absolutely protect you from having your program copied? Of course not! Anybody with the same skills

and training as you (for instance, in machine language) will still be capable of eventually figuring things out if that is their ultimate goal. The point now is, if they are so skilled and cunning, why would they bother? Chances are they would rather be devoting their time and energy to producing their own original creations.

Notice that protection of a program (preventing it from being copied or duplicated) is a different matter than securing an individual PC from unauthorized use. The former problem assumes that the program must be distributed by some practical means (tape or module). The latter assumes that the program is installed in the PC and that the use of that particular PC (and hence the program(s) it contains) is what is being secured. Several people have submitted programs to *PCW* that are claimed to provide protection along these lines for the PC-1500. Perhaps, if reader interest warrants, we can provide a sample of this type of security program in a future issue?

Code Breaking

A number of readers have asked for information on how to approach deciphering the machine code for a computer. That is, how did *PCW*'s authors break the code for the LH-5801 CPU? We suspect that much of the recent expressions of interest in this matter have something to do with people wanting to work on discovering the machine language of some of the new PCs!

Whatever the reason for the interest, here are a few comments on the subject:

First, work on mapping memory. Determine which sections are devoted to RAM and ROM. Perform experiments (using BASIC or whatever language is available on the machine) to further define how RAM is used. That is, what addresses are used for program storage, where variables are stored, what parts of RAM appear to be used as "system" resources. This type of information should be arranged in orderly fashion, such as by memory address value.

Next, print out a dump of ROM memory. Preferably, each line of the dump should show raw machine code (in whatever number base you prefer) and the character produced by ASCII values. Using this technique can help you pick out various kinds of lookup and conversion tables within the ROM.

Then, obtain a histogram on the contents of ROM. That is, for each possible value (0 - 255 in an 8-bit machine), find out how many times the value occurs throughout ROM. It is fairly easy to do this using a BASIC program that counts the number of occurrences for each possible value and stores this in a 256-element array.

From this point on, start analyzing the data while calling on your own personal knowledge. The more experience you have in terms of computer science or mathematics, the higher the chances for immediate success. Use any data that has relevance for you. For instance, when Norlin Rober first started working on the PC-1500 ROM, he (being a skilled mathematician) was quickly able to spot values used to make mathematical conversions and tables used to calculate mathematical functions within the ROM.

Start making assumptions about the ROM code and playing hunches. For instance, in a large machine language program such as a BASIC interpreter, chances are good that some of the most frequently used instructions will be "loads" or "stores," "jumps," "branches" or "calls" to subroutines and "return" instructions. Use the histogram you made of ROM, the dump of memory, and your knowledge of the use of addresses in RAM to try and put some of your hypothesis together. For instance, you might assume that most call and jump instructions would be to locations within the ROM. This means that the call or jump opcode would be followed by address bytes having values within the range utilized by ROM. Take the values having the ten highest occurrences from the histogram of the ROM code that you compiled. Find occurrences of those values within ROM and see if any of them appear to be followed by reasonable address values. If so, mark the memory dump at those locations. When you get a

handful of prospective jump or call locations, start examining the code in the vicinity of those addresses. See if you find the same code value immediately preceding each prospective jump or call address. Any luck? If so, you may have found the opcode for a subroutine "return" instruction!

On a machine such as the PC-1500, once a solid "guess-estimate" had been made on an opcode such as return, it was easy to confirm the finding. All that had to be done was POKE the suspected return opcode into a specific address in RAM and then attempt a CALL (using the BASIC statement) to that location. Finding that the PC immediately returned from the machine mode when the CALL was executed (from BASIC), regardless of where it was placed in RAM, quickly confirmed the tentative hypothesis. Once the thrill of confirming your first "find" has subsided, you can continue your investigations in a number of ways.

One way is to look for code in ROM that contains RAM addresses. Chances are good these involve instructions that

are "pointing" to RAM. These may be "loads" or "stores." Try CALLing a ROM address that you have potentially identified as the entry point to a routine or subroutine. Ideally, the routine(s) you are trying at this point contain code that points to RAM locations. Observe what happens to these RAM locations. (Note, be prepared to do a lot of system resetting. You are likely to "bomb" the PC on almost every trial at this stage in the sleuthing process!) Do the RAM location(s) change? If so, you may be executing some "store" directives. Keep a record of your suspicions, correlate your findings with the ROM histogram and memory dump, try to confirm each hypothesis by repeating the test at different locations in ROM (that have similar opcode/data patterns).

It is great fun for some people. Time-consuming, that is for sure, but it can be exciting. How much work to break the code for a CPU such as the LH-5801? Perhaps as much as 300 to 400 hours to find just about all the directives. It is sort of like working on a giant puzzle. Bring on the PC-1350!

FOR PC-1350 USERS

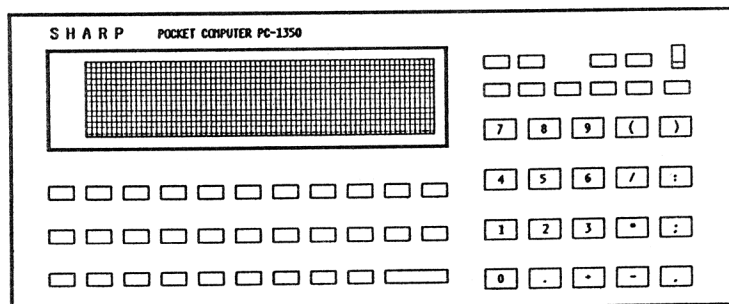
WELCOME ABOARD

Be sure to read the review of the PC-1350 elsewhere in this issue of *PCW*. This is the first issue having a column specifically devoted to this unit, so let's take a look at some of the most exciting news relating to this machine.

First of all, the BASIC language installed in the PC-1350 is virtually identical to that used in the PC-1260/61 models. Thus, the BASIC programs listed under that section (in *PCW*) should work without modification. So, presto! You can use the payroll and memory dump programs that are in this issue on your new PC-1350. Indeed, the compatibility between the 1260 and 1350 is so good that you can load programs from tapes made with the 1260 directly into the 1350. (You cannot, however, go the other way. As the Sharp manual points out, the PC-1350 is "upwards" compatible. You can load from an earlier model into the new 1350. You cannot use tapes to go the other way.)

Programs designed for the operation on the PC-1250 (also the Radio Shack PC-3) and even for the original PC-1211 (also the Radio Shack PC-1) will work in the new 1350 with little or no alteration. (Unless you used programming tricks such as implied multiplication or left off closing parenthesis. In such cases, you will have to do some editing.)

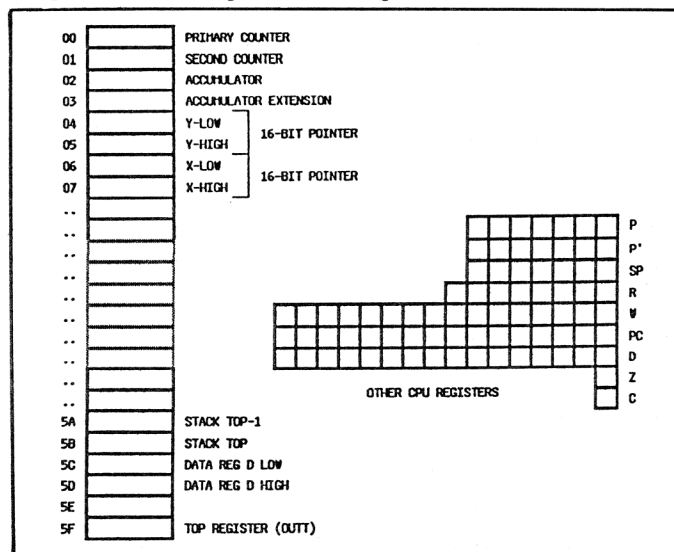
Now for the next hot piece of news. Preliminary investigation indicates that the PC-1350 utilizes the same CPU as that in the PC-1250 and PC-1260 series! And wouldn't you know it, a number of *PCW* fans have been hard at work busting the PC-1250 machine language code. In fact, next issue we will be publishing a comprehensive report on the work of a new CPU sleuth, *Rick Wenger*, who has pretty well decoded the instruction set for the CPU used in the 1250 (and consequently that of the 1260 and, it appears, the PC-1350).



If you have been doing any exploring on your own, an accompanying diagram summarizes the internal registers associated with the unit's CPU. You can use the memory dump program described for the 1250 to begin mapping out the 1350. It appears that there is some RAM in the 1350 beginning at hexadecimal address 66000. This should be a fun machine to explore. Imagine what kind of machine language programming tools you could tuck into one of those 16K RAM cards?

The PC-1350 is considerably easier to program from the keyboard than other Sharp models. This is primarily due to the re-arranged keyboard which eliminates having to use the shift key in order to input a lot of the punctuation marks used in BASIC programming. Symbols such as the comma, colon, semicolon, etc., have their own separate keys. Also, the 4-line display, capable of displaying 96 characters at a time, makes it a lot easier to review and edit programs. Editing is also aided by the fact that you can insert or delete characters using special keys for those operations. No more constant use of the extra shift key.

Diagram Probable Layout of CPU Registers in the PC-1350.



FROM THE WATCH POCKET

The November, 1985, issue of McGraw-Hill's *BYTE* magazine has an interesting review (beginning on page 416) of a new pocket computer being manufactured in England. It is known as the Psion Organiser. The article was interesting for several reasons: the pure "hype" it contained, the author's apparent ignorance about the design and operation of other pocket computers that have been on the market for years, and the fact that the Psion Organiser uses a concept that seems readily applicable to other pocket computers, particularly models similar to the Sharp PC-1350.

The Psion Organiser has apparently been designed primarily as a data-processing device. Its claim to fame seems to be based on the concept of having its data base stored in a programmable read-only memory (PROM). Up to two (8K or 16K) PROM chips can be installed at a time. As the user creates a data base (such as a name and address file or telephone list), the PROM is blasted. This is a one-shot deal. Once a file is filled, you cannot re-program it unless you take the chip out and return it to a dealer. You can apparently, however, remove an entry from a file (probably by having the entry blasted to an all-zeroes condition) so that it is no longer processed by the program.

This is a nice concept that seems readily implementable on other PCs. Why couldn't Sharp make low-cost PROM cards for the PC-1350? PROMs are cheap. (Psion sells 16K versions for about \$20.00.) The availability of such cards would allow users to create and maintain low cost data bases. Software vendors could also use such PROM cards to distribute application packages. Low cost ultra-violet erasers can be used to erase such PROMs so that they can be used again. This

is a practical, feasible component.

But, whoever does it, let's hope they don't lay down the kind of hype that apparently surrounds the Psion machine. The *BYTE* article likens the tiny PROMs to serially accessed disk drives. That seems a pretty far-fetched comparison.

The *BYTE* article also contains other questionable implications. It alludes to the fact that the Psion machine contains a "proper" 8-bit microcomputer and states that pocket computers such as the Sharp PC-1500 are just "the modern version of the programmable calculator." It seems the author may be somewhat misinformed. The LH-5801 CPU inside the PC-1500 is certainly just as "proper" an 8-bit computer as the HD6301X touted for the Psion Organiser. The Psion unit is said to be "based on a very fast search algorithm with partial word matching" and to be capable of searching a 16K database in a mere five seconds. Wow! Incredible, what? Not really. Certainly machine language implementations on a PC-1500 or PC-1350 could perform just as well.

So what is the point of this discussion? Merely to point out that in England there is a company that thinks it can make a business out of putting a microcomputer into a pocket-sized package and selling its "database" capabilities. It need not have gone to all the trouble of designing such a hardware package. It could have simply added such capabilities to a PC-1500 or PC-1350. Seems the door for someone else doing this is still wide open! If Sharp can't see their way to doing it, perhaps some other enterprising firm will take a crack at it!

Speaking of Sharp. How about that new PC-1350? It seems a pretty nice machine, with some notable improvements over the PC-1500. But, the left off a critical capability: there is no real-time clock! Shucks. There you are with a nice built-in serial communications port, a wonderful 4-line display, a powerful BASIC package, and no ability to perform real-time operations! No way to automatically turn the unit on and have it take measurements of its own accord. No means of recording the time of events or sounding an alarm to remind you of appointments. What a waste. Sharp blew a lot of the potential engineering and scientific market by leaving out this critical capability. They need to do some more market research about their potential users. Perhaps they should take a gander at Hewlett-Packard. Good ole HP wouldn't blow the opportunity to have their PCs hooked up as controllers and recorders by leaving out a real-time clock! The PC-1350 has plenty to offer, but it still ain't perfect.

If you are a businessperson who schedules a lot of appointments and does a lot of travel, you may be interested in a PC called the PAD. This is a special-purpose pocket computer. It will retain up to 80 appointments and memo notes in its 4K of memory. It *does* have a *real-time clock* so that it can alert you to an appointment by giving an alarm. There is also a section that can be used to store expenses in 8 categories. And, it can maintain an alphabeticized list of phone numbers. The PAD weighs just 5 ounces, measures 3-1/2 by 5-1/4 by 3/8 inches, and runs for about 8 months on 3 silver oxide batteries. It is supplied with a vinyl case (that contains extra room for holding credit cards while traveling) and retails in the U.S. for \$99.00. For additional information contact: The Sharper Image, P.O. Box 26823, San Francisco, CA 94126-6823, and ask for literature on their Electronic Memo Pad, product number ABX139.

1985 is going to be an exciting year for pocket computer aficionados. Look for some useful peripherals to come out in support of the PC-1350. A better mass storage device (than audio tape) is certainly realistic now. How about a battery powered floppy disk drive? That seems to be a possibility in the not-too-distant future. How about a miniature modem? A color printer? Say, how about some cables with connectors that mate to the PC-1350's serial port? That would help the "do-it-yourselfers" interface the new PC to "what-have-you" while the manufacturers decide what they are going to build.

Available Only by Prepaid Subscription for a Calendar Year Period (January - December). You are sent back issues for the calendar year to which you subscribe, at the time you enroll.

- Enroll me as a 1985 Subscriber (Issue numbers 37-44). \$24.00 in U.S. (U.S. \$30.00 to Canada/Mexico. Elsewhere U.S. \$40.00 payable in U.S. funds against a U.S. bank.)
- Enroll me as a 1984-85 Subscriber (Issue numbers 31-44). \$42.00 in U.S. (U.S. \$51.00 to Canada/Mexico. Elsewhere U.S. \$70.00 payable in U.S. funds against a U.S. bank.)
- Enroll me as a 1983-85 Subscriber (Issue numbers 21-44). \$78.00 in U.S. (U.S. \$93.00 to Canada/Mexico. Elsewhere U.S. \$120.00 payable in U.S. funds against a U.S. bank.)
- Enroll me as a 1982-85 Subscriber (Issue numbers 11-44). \$102.00 in U.S. (\$125.00 to Canada/Mexico. Elsewhere U.S. \$160.00 payable in U.S. funds against a U.S. bank.)
- Check here if paying by MasterCard or VISA. Please give credit card information below.

Orders must be accompanied by payment in full.
All checks must be magnetically encoded, payable in U.S. funds and drawn against a U.S. bank.

Name: _____

Addr: _____

City: _____ State: _____ Zip: _____

MC/VISA #: _____

Signature: _____ Exp. Date: _____

mail this order form to:

POCKET COMPUTER NEWSLETTER

P.O. Box 232, Seymour, CT 06483